National University of Computer and Emerging Sciences



# Lab Manual 13

**Object Oriented Programming – CL1004**

|  |  |
| --- | --- |
| Course Instructor | Dr. Saira Karim |
| Lab Instructor(s) | Ms. Amna Zulfiqar Mr. Muhammad Adeel |
| Section | BCS-2B |
| Semester | Spring 2023 |
| Date | 13-05-2023 |

Department of Computer Science FAST-NU, Lahore, Pakistan

# Lab Manual 13

**Important Note:**

* **You may find the syntax to accomplish these exercises from lecture demo.**
* **Add Necessary Comments in you code to justify your logic.**
* **Comment exercise number or statement at the start of your code**
* **Save each exercise in .cpp file with your roll no, ex and lab number e.g.**
* **22LXXXX\_EX01\_Lab01.cpp**
* **Place all of your exercises in a folder a Zip it (Do not create .rar file) with roll no and lab no. e.g. 22LXXX\_Lab01.zip**
* **Make sure that the interface of your program is user friendly i.e. properly display information.**
* **Properly follow the coding standards.**

**What is Exception Handling?**

An exception is a problem that arises during the execution of a program. A C++ exception is a response to an exceptional circumstance that arises while a program is running, such as an attempt to divide by zero.

Exceptions provide a way to transfer control from one part of a program to another. C++ exception handling is built upon three keywords: **try, catch,** and **throw**.

* **throw** − A program throws an exception when a problem shows up. This is done using a **throw** keyword.
* **catch** − A program catches an exception with an exception handler at the place in a program where you want to handle the problem. The **catch** keyword indicates the catching of an exception.
* **try** − A **try** block identifies a block of code for which particular exceptions will be activated. It's followed by one or more catch blocks.

Assuming a block will raise an exception, a method catches an exception using a combination of the **try** and **catch** keywords. A try/catch block is placed around the code that might generate an exception. The syntax for using try/catch as follows

|  |
| --- |
| try {  // code }  catch( ExceptionName e1 ) {  // catch block }  catch( ExceptionName e2 ) {  // catch block }  catch( ExceptionName eN ) {  // catch block } |

Throwing Exceptions

Exceptions can be thrown anywhere within a code block using throw statement. The operand of the throw statement determines a type for the exception and can be any expression and the type of the result of the expression determines the type of exception thrown.

Following is an example of throwing an exception when dividing by zero condition occurs:

|  |
| --- |
| double division(int a, int b) {  if( b == 0 ) {  throw{ "Division by zero condition!"};  }  return (a/b);  } |

**Exercise 1: Inheritance, Polymorphism and Exception Handling:**

Create a C++ program that simulates a banking system. The program should have two types of accounts: checking accounts and savings accounts. Users should be able to create any type of account and deposit or withdraw from it. Use exception handling to catch and handle any errors that may occur during the transaction (Error in withdrawal transaction occurs when the accounts have insufficient balance or you withdraw more than a limit i.e. 25000 in this case.).

1. Create an **Abstract Account** Class with the following functions:

double get\_balance()

void deposit(double amount)

void withdraw(double amount) = 0;

~Account() {}

1. Create the **CheckingAccount** and **SavingsAccount** Classes that inherit from Account Class. They should have additional private member “double balance” that must be initialized with the creation of the account.
2. Create **AccountException** Class to and to catch and handle any errors that may occur during the transaction.
3. Drive the program in main(). Ask the user to enter account type and initial amount for account creation. Then ask the user to Enter a transaction amount. If the transaction amount is invalid or exceeds the limit an exception should be raised and a proper message should be displayed.